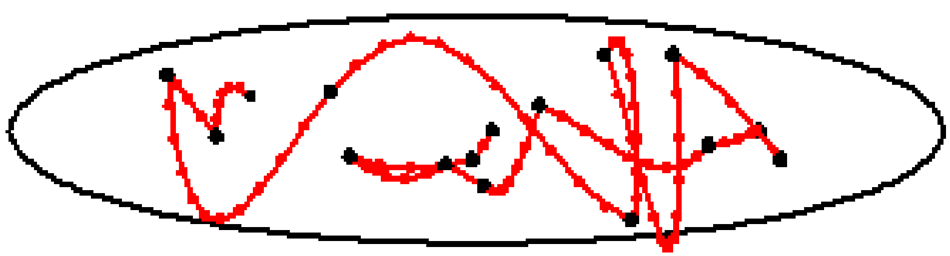
Use of probability gradients in hybrid MCMC and a new convergence test

Ken Hanson

Methods for Advanced Scientific Simulations Group



This presentation available under http://www.lanl.gov/home/kmh/

Acknowledgements

- MCMC experts
  - Dave Higdon, Frank Alexander, Julian Besag, Jim Guberantus, John Skilling, Malvin Kalos, Radford Neal
- General discussions
  - Greg Cunningham, Richard Silver

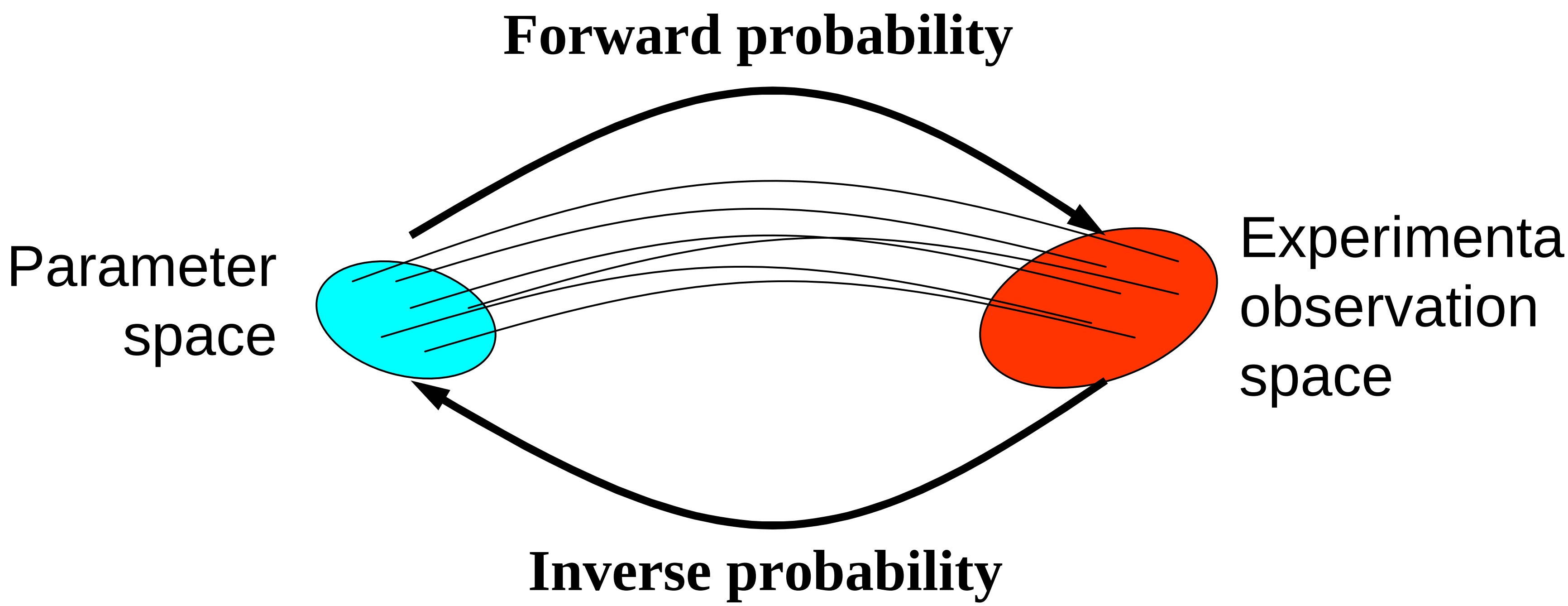
Outline

- Setting - Bayesian inference for simulation codes
  - numerous continuous parameters
  - expensive function evaluations
- Adjoint differentiation on basis of code
  - efficiently calculates gradient of computed scalar quantity
  - uses
  - methods of implementation
- Hybrid Markov Chain Monte Carlo
  - basic algorithm
  - requires gradient of minus-log-probability
- A method to test convergence of MCMC sequence
  - based on gradient of minus-log-probability

Uses of adjoint differentiation

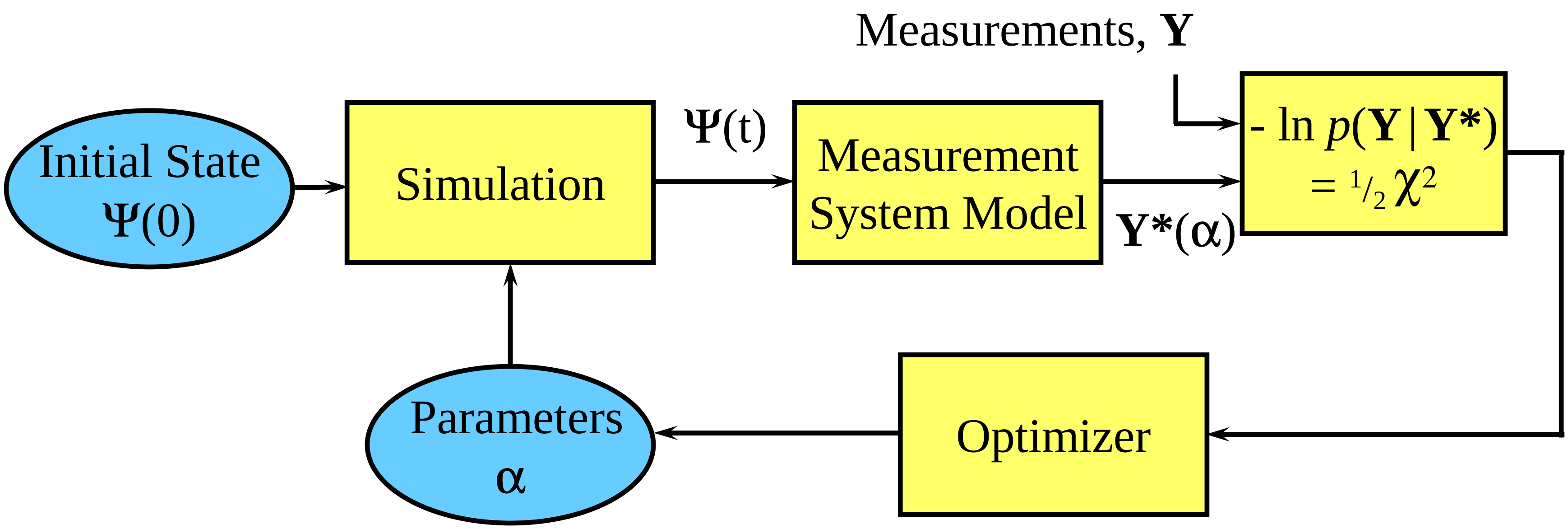
- Fitting large-scale simulations to data (regression):
  - atmosphere and ocean models, fluid dynamics, hydrodynamic
- Reconstruction: imaging through diffusive media
- Hybrid Markov Chain Monte Carlo
- Uncertainty analysis of simulation code
  - sensitivity of uncertainty variance to each contributing cause
- Metropolis-Hastings MCMC calculations
  - sensitivity of efficiency (or acceptance fraction) wrt proposal distribution parameters

Forward and inverse probability



- Forward probability - determine uncertainties in observables resulting from model parameter uncertainties
- Inverse probability - infer model parameter uncertainties from uncertainties in observables; **inference**

Maximum likelihood estimation by optimzation

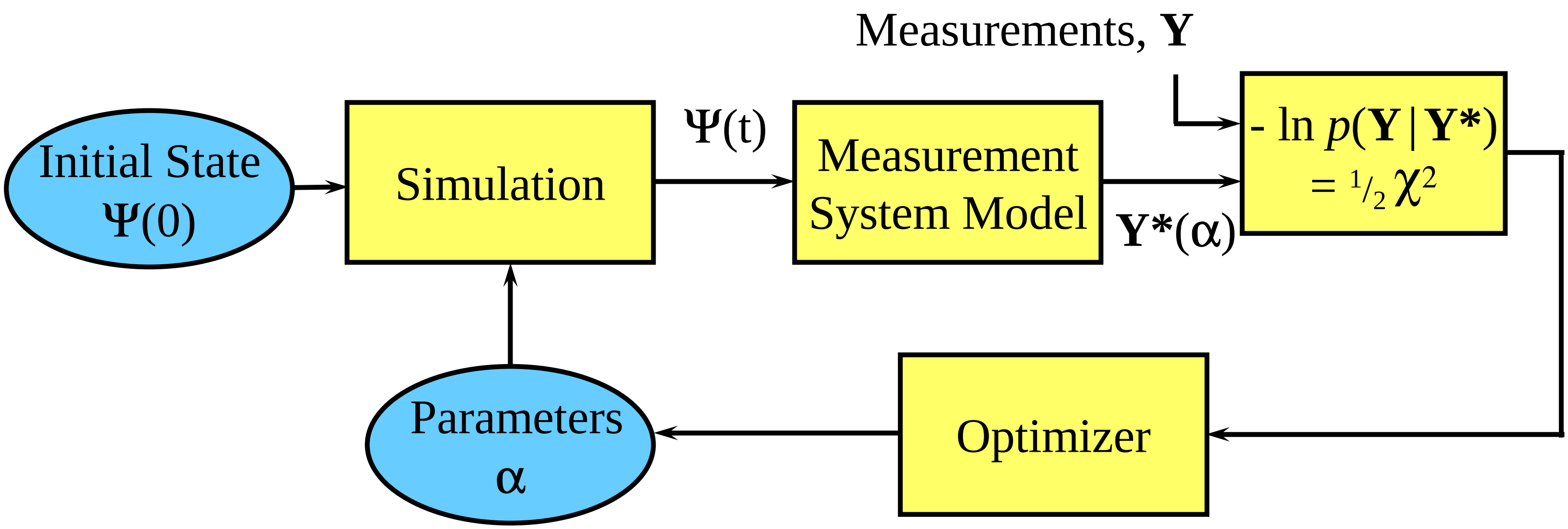


- Find parameters (vector α) that minimize

$$-\ln p(\mathbf{Y}|\ \mathbf{Y^*}(\alpha)) = \frac{1}{2}\chi^2 = \sum \frac{(y_i - y_i^*)^2}{\sigma_i^2}$$

- Result is **maximum likelihood estimate** for α
  - also known as minimum-chi-squared or least-squares solution
- Prior information used to overcome ill-posedness; Bayesian approach

Maximum likelihood estimation by optimization



- Find minimum in $-\ln p(\mathbf{Y} \mid \mathbf{Y}^*(\alpha)) = \frac{1}{2}\chi^2$ by iteration over parameters α
- Optimization process is accelerated by using **gradient-based algorithms**; therefore need gradients of simulation and measurement processes
- **Adjoint differentiation** facilitates efficient calculation of gradients, i.e. derivative of scalar output ($\frac{1}{2}\chi^2$) wrt parameters α

Derivative calculation by finite differences

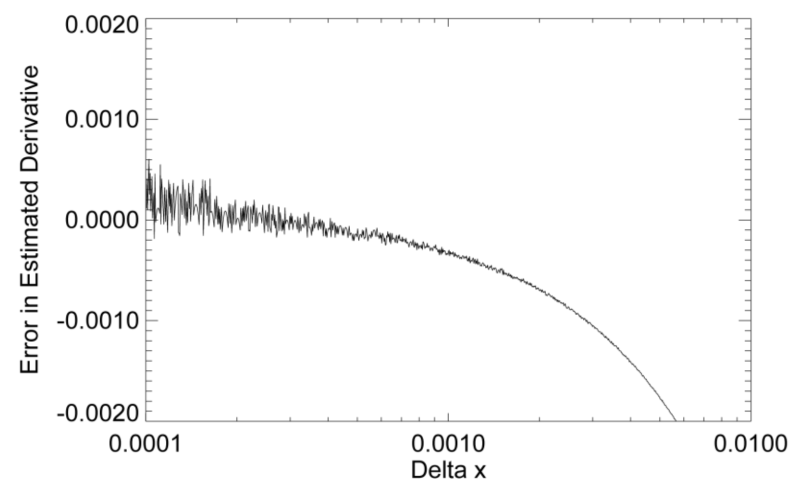
- Derivative for function defined as limit of ratio of finite differences:

$$\left.\frac{df}{dx}\right|_{x1} = \lim_{\Delta x \to 0} \frac{f(x1+\Delta x) - f(x1)}{\Delta x}$$

  Wish to estimate derivatives of calculated function for which there is no analytic relation between outputs and inputs

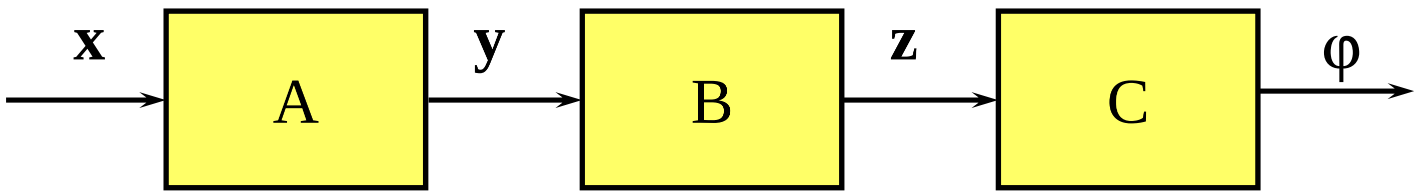
- Numerical estimation based on finite differences is problematical:
  - difficult to choose perturbation Δx
  - **# function evaluations ~ # variables**

- Estimation based on functionality implied by computer code is more reliable



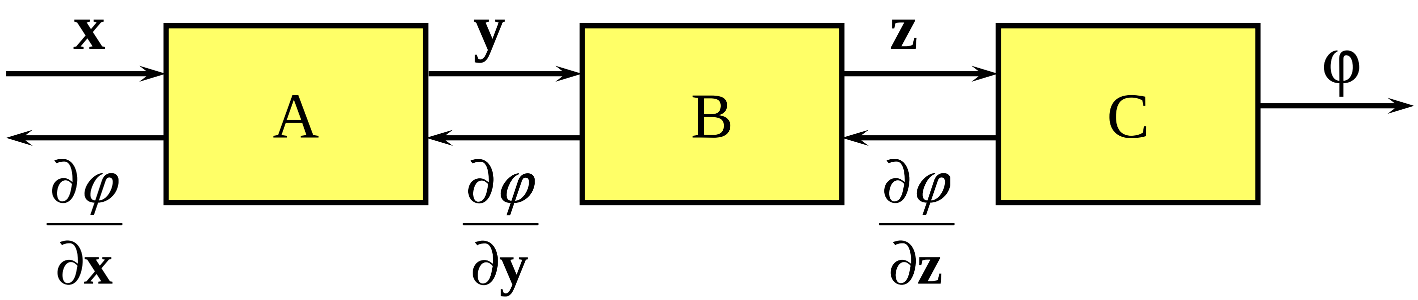
Error in derivative of sin(x) vs. Δx at $x = \pi/4$

Differentiation of sequence of transformations



- Data-flow diagram shows sequence of transformations A->B->C that converts data structures **x** to **y** to **z** and to scalar φ (**forward calculation**)
- Desire derivatives of φ wrt all components of **x**, *assuming* that φ is *differentiable*
- Chain rule applies: $$\frac{\partial \varphi}{\partial x_i} = \sum_{j,k} \frac{\partial y_j}{\partial x_i} \frac{\partial z_k}{\partial y_j} \frac{\partial \varphi}{\partial z_k}$$
- Two choices for summation order:
  - doing *j* before *k* means derivatives follow data flow (forward calculation)
  - doing *k* before *j* means derivatives flow in reverse (adjoint) direction

Adjoint Differentiation In Code Technique ADICT



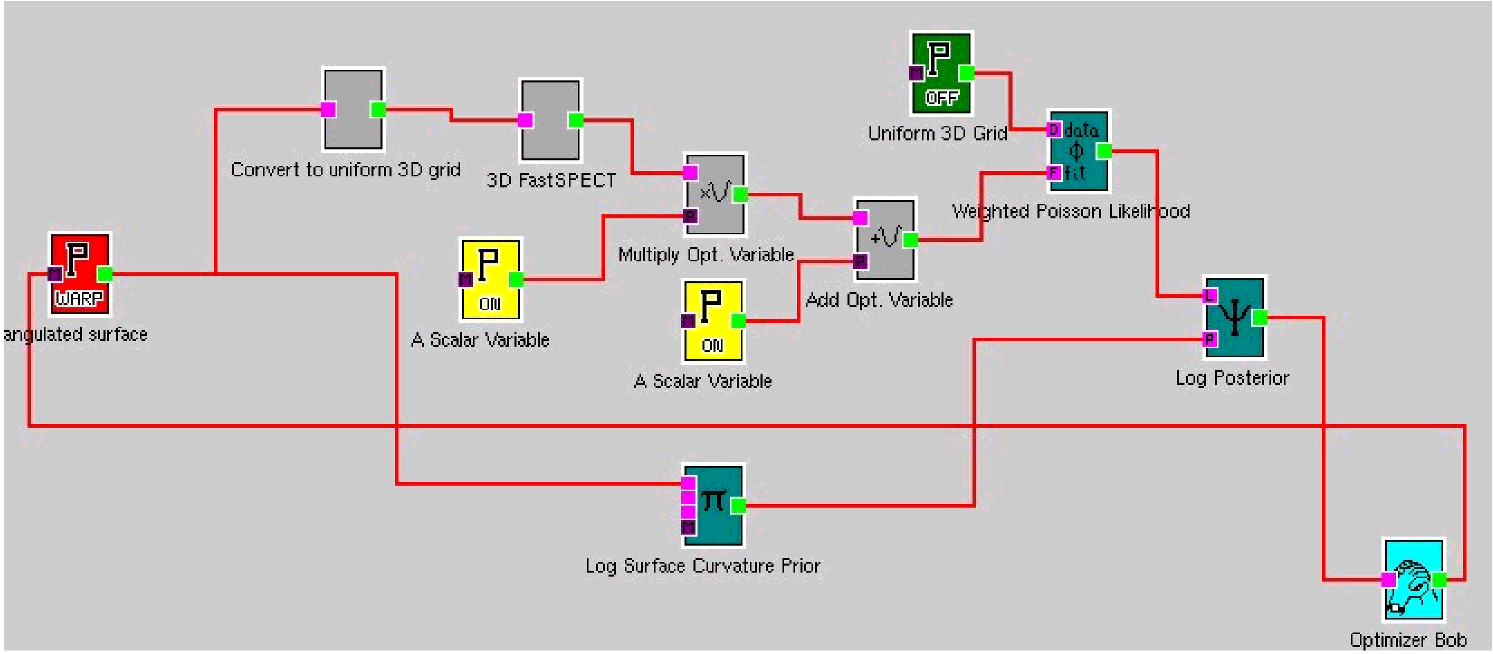
- For sequence of transformations that converts data structure **x** to scalar φ
- Derivatives $\frac{\partial \varphi}{\partial \mathbf{x}}$ are efficiently calculated in the reverse (adjoint) direction
- **Code-based approach**: logic of adjoint code is based explicitly on the forward code or on derivatives of the forward algorithm
- **Not** based on the theoretical eqs., which forward calc. only approximates
- Only assumption is that φ is a **differentiable function** of **x**
- CPU time to compute **all** derivatives is comparable to forward calculation

Level of abstraction of implementation

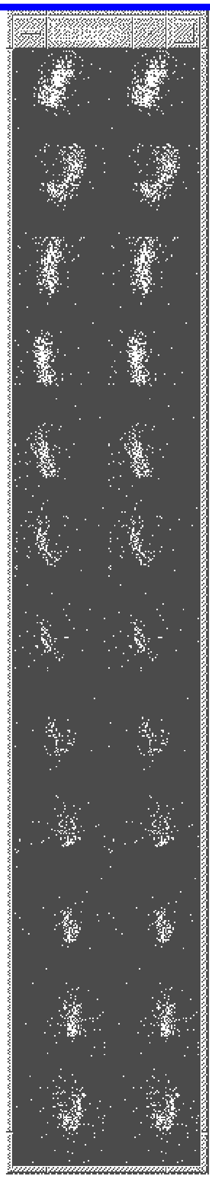
- One can choose to differentiate forward calculation at various levels of abstraction
  - model based
    - e.g., differentiate partial differential equations and solve
    - not advised because forward codes only approximates model
  - module or algorithm based
    - differentiate each basic algorithm (Bayes Inference Engine)
  - code based
    - direct interpretation of computer code (FORTRAN, C, etc.)
    - automatic differentiation utilities produce derivative code(s)
  - instruction based
    - reverse the sequence of CPU instructions for any particular calc.

Example of algorithm-based approach

- Bayes Inference Engine (BIE) created at LANL
  - modeling tool for interpreting radiographs
  - BIE programmed by creating data-flow diagram linking transforms, as shown here for 3D reconstruction problem
- **Adjoint differentiation crucial to BIE success**



3D reconstruction



Data

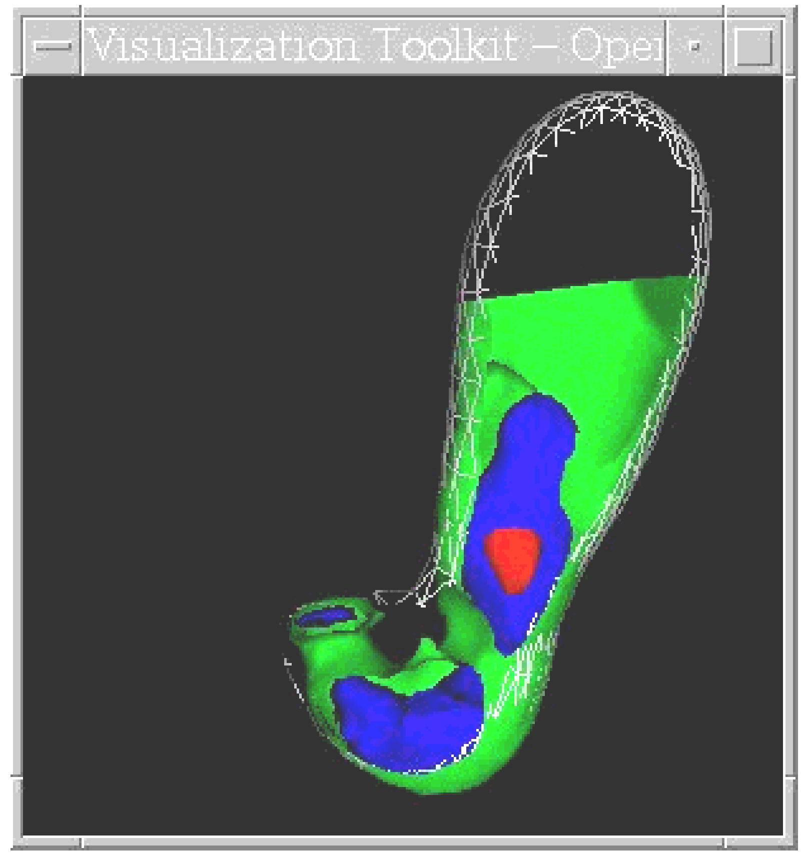
Frame 51 out of 100

50 ms, 4000 counts

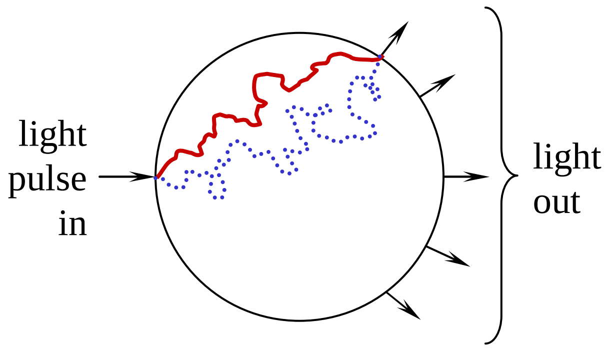
24 views

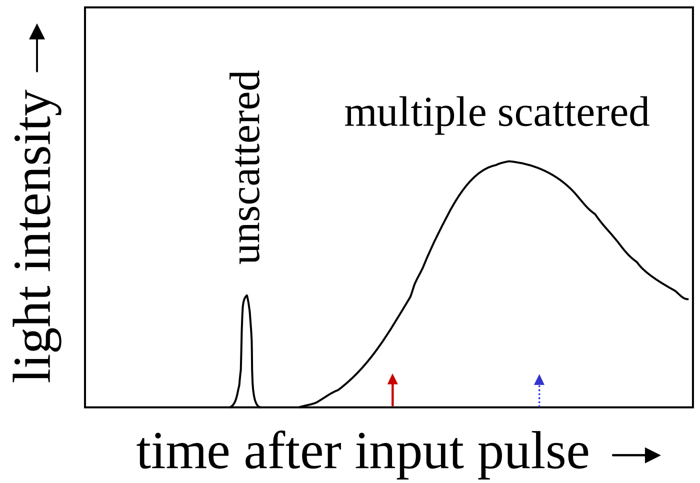
Reconstruction

Variable intensity inside deformable boundary



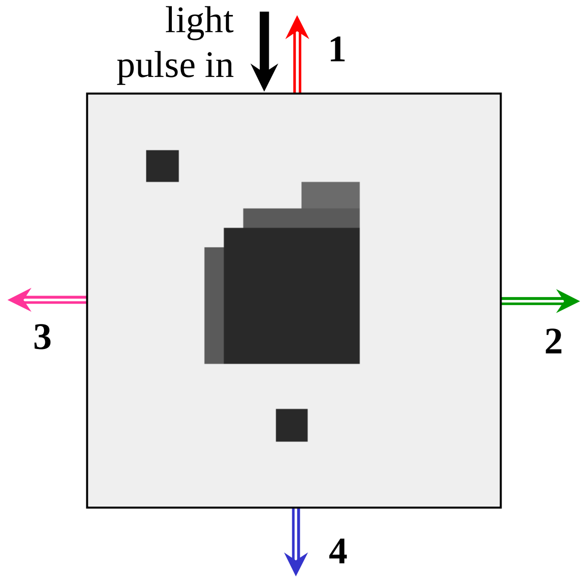
Simulation of light diffusion in tissue



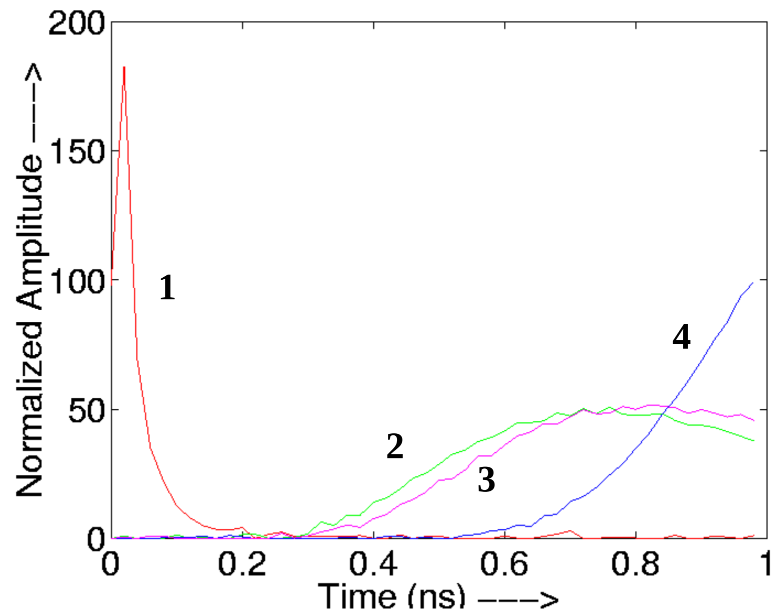


- IR light photons in broad, retarded peak literally "diffuse" by multiple scattering from source to detector
  – time is equivalent to distance traveled
  – diffusion equation models these multiply-scattered photons
  – these photons <u>do not</u> follow straight lines

Optical tomography – invert diffusion process



$0.7 < D < 1.4$ cm^{2}ns^{-1} (μ_a=0.1 cm^{-1})



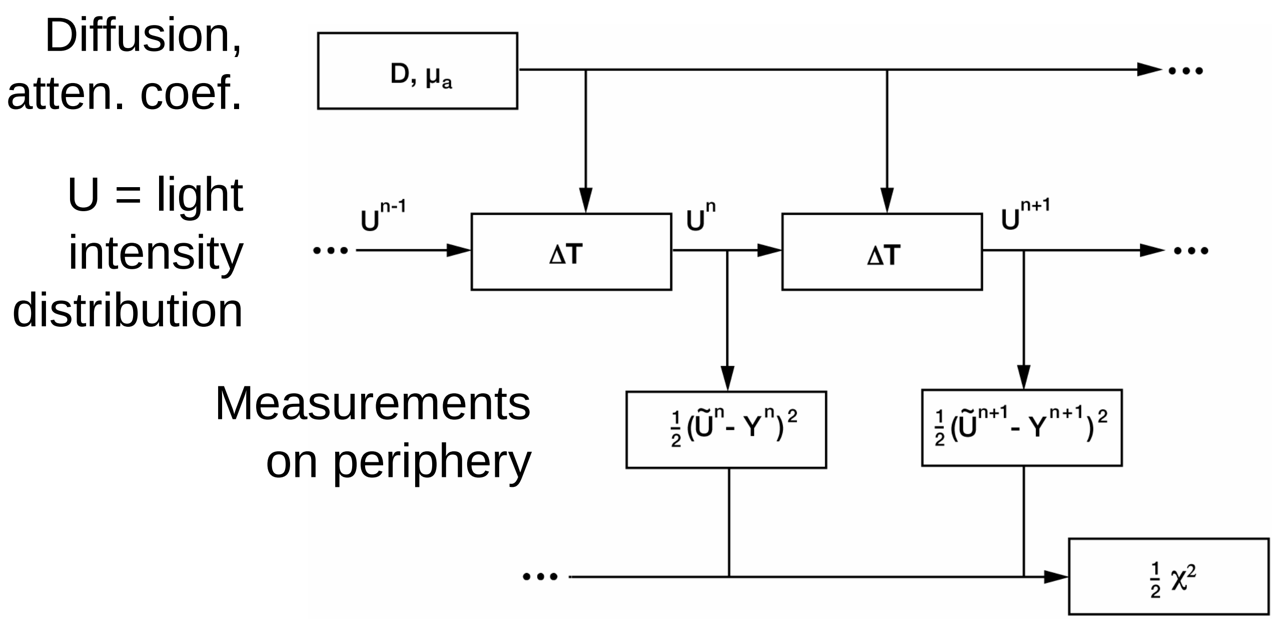
– for assumed distribution of diffusion coefficients (left)

– predict time-dependent output at four locations (right)

– reconstruction problem - determine image on left from data on right

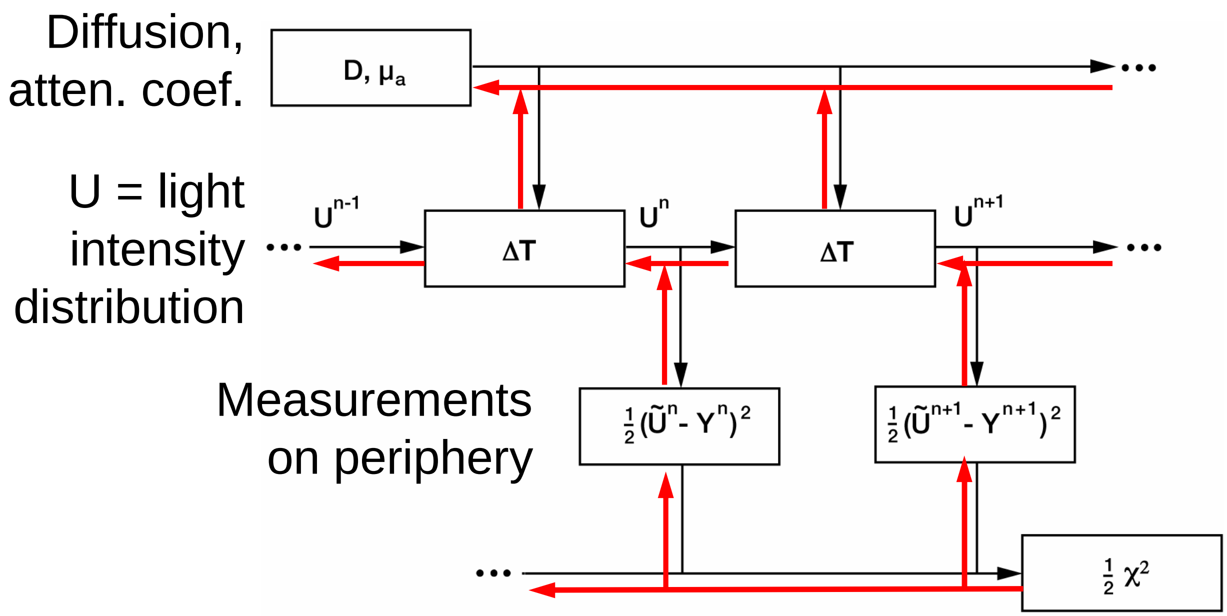
Finite-difference calculation

- Data-flow diagram shows calculation of time-dependent measurements by finite-difference simulation
- Calculation marches through time steps Δt
  - new state $\mathbf{U}_{n+1}$ depends only on previous state $\mathbf{U}_n$



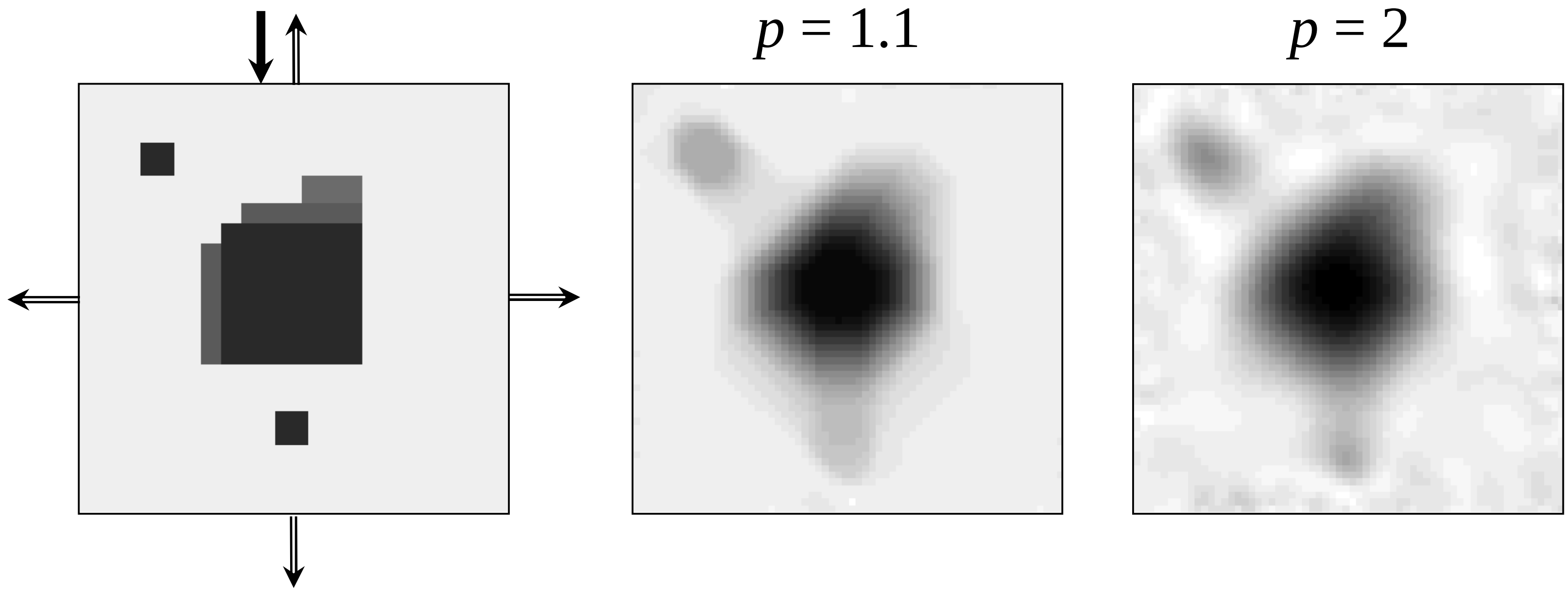
Adjoint differentiation in diffusion calculation

- Adjoint differentiation calculation precisely reverses direction of forward calculation
- Each forward data structure has an associated derivative
  - where $\mathbf{U}_n$ propagates forward, $\dfrac{\partial \varphi}{\partial \mathbf{U}_n}$ goes backward $(\varphi = \frac{1}{2}\chi^2)$



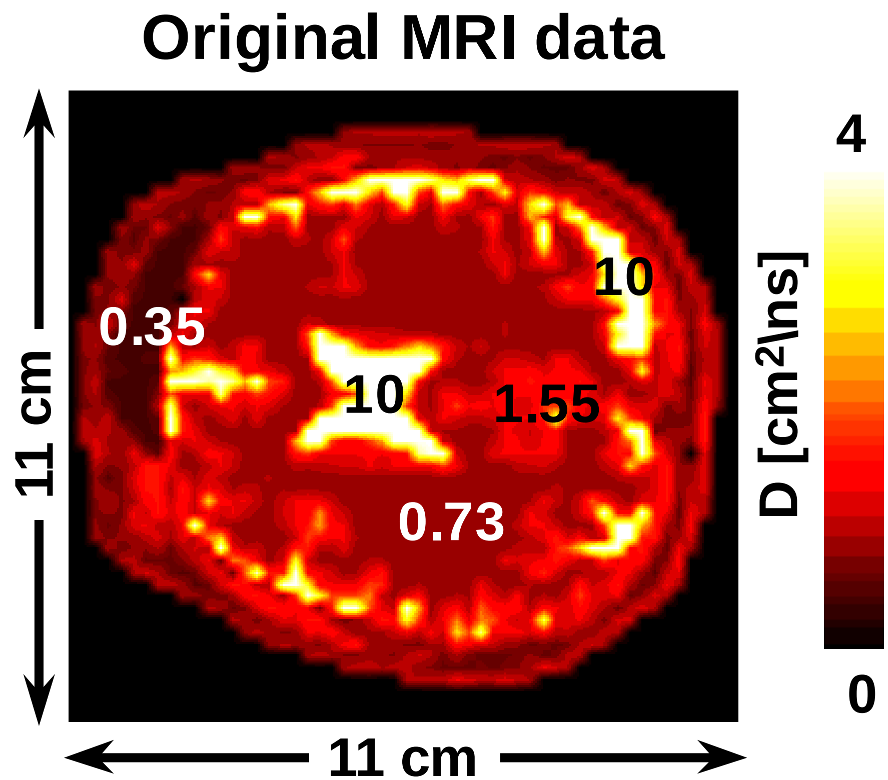
Optical tomographic reconstruction – determine image of light diffusion characteristics from measurements of how will IR light passes through tissue

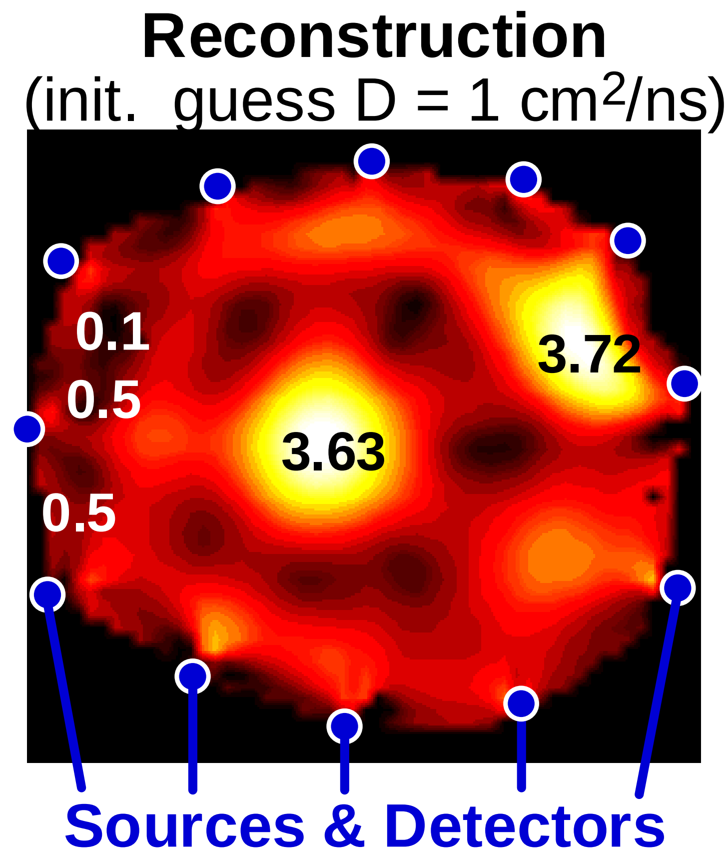
Reconstruction of simple phantom



- Measurements
  - section is $(6.4\text{cm})^2$, $0.7 < D < 1.4$ cm^{2}ns^{-1} ($\mu_{abs} = 0.1$ cm^{-1})
  - 4 input pulse locations (middle of each side)
  - 4 detector locations; intensity measured every 50 ps for 1 ns
- Reconstructions on 64 x 64 grid from noisy data (rmsn = 3%)
- Conjugate-gradient optimization algorithm

Reconstruction of Infant's Brain I





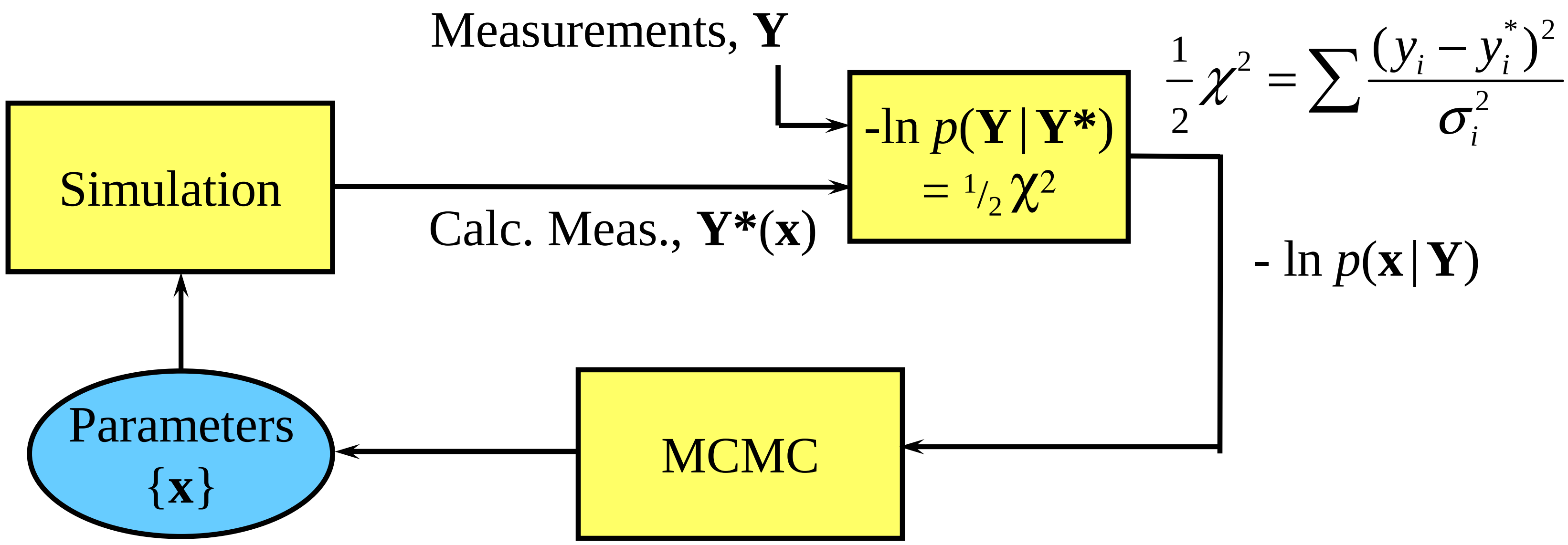
hematoma (left side) and cerebrospinal fluid pocket (upper right)

(60 iterations ~ 70 min)

Automatic differentiation tools

- Several tools exist for automatically differentiating codes; various capabilities, e.g., forward or reverse (adjoint) differentiation, handling of large codes, etc.
  - FORTRAN 77 (90 under development)
    - ADIFOR (reverse mode)
    - TAMC (reverse mode)
    - TAPENADE (reverse mode)
  - C (C++ under development)
    - ADIC
    - ADOL-C (reverse mode)
  - MATLAB
    - ADMAT
- Very active area of development

MCMC for simulations



- - log(likelihood) distribution is result of calculation; function of parameters **x**
- Markov Chain Monte Carlo (MCMC) algorithm draws random samples of **x** from posterior probability $p(\mathbf{x}|\mathbf{Y})$
- Produces plausible set of parameters {**x**}; therefore model realizations

MCMC - problem statement

- Parameter space of n dimensions represented by vector $\mathbf{x}$
- **Draw a set of samples** $\{\mathbf{x}_k\}$ from a given "arbitrary" **target probability density function** (pdf), $q(\mathbf{x})$
- **Only requirement** typically is that one **be able to evaluate** $Cq(\mathbf{x})$ for any given $\mathbf{x}$, where C is an unknown constant; that is, $q(\mathbf{x})$ need not be normalized
- Although focus here is on continuous variables, MCMC applies to discrete variables as well

Uses of MCMC

- Permits evaluation of the expectation values of functions of $\mathbf{x}$, e.g.,

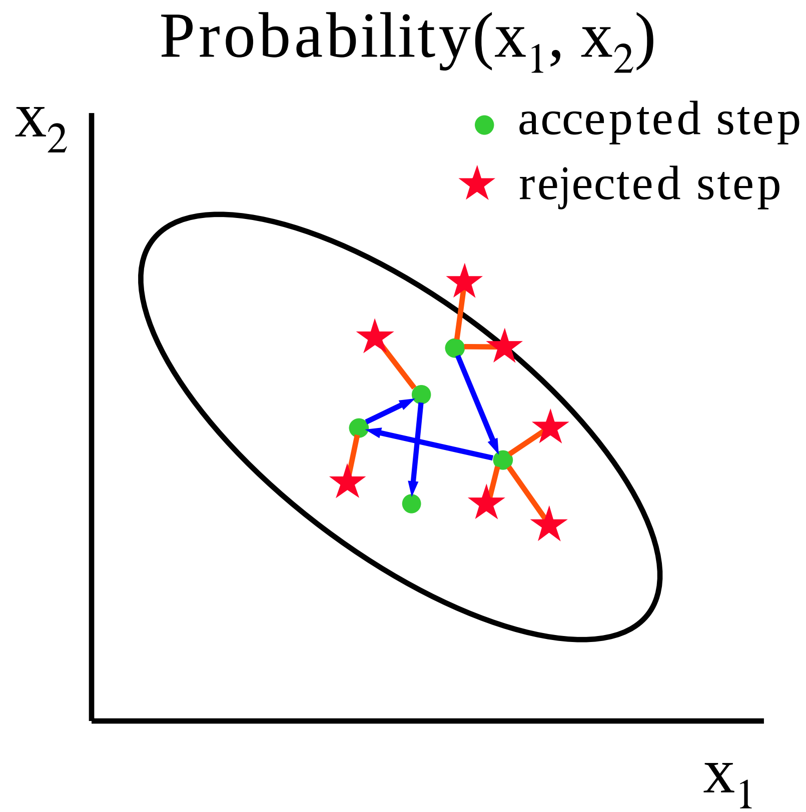
$$\langle f(\mathbf{x})\rangle = \int f(\mathbf{x})\, q(\mathbf{x})\, d\mathbf{x} \cong (1/\mathrm{K})\, \Sigma_{\mathrm{k}}\, f(\mathbf{x}_{\mathrm{k}})$$

  – typical use is to calculate mean $\langle\mathbf{x}\rangle$ and variance $\langle(\mathbf{x} - \langle\mathbf{x}\rangle)^2\rangle$
- Useful for evaluating integrals, such as the partition function for properly normalizing the pdf
- Dynamic display of sequences provides visualization of uncertainties in model and range of model variations
- Automatic marginalization; when considering any subset of parameters of an MCMC sequence, the remaining parameters are marginalized over (integrated out)

Metropolis Markov Chain Monte Carlo

Generates sequence of random samples from an arbitrary probability density function

- Metropolis algorithm:
  - draw trial step from symmetric pdf, i.e., $t(\Delta\mathbf{x}) = t(-\Delta\mathbf{x})$
  - accept or reject trial step
  - simple and generally applicable
  - requires only calculation of target pdf, $q(\mathbf{x})$, for any $\mathbf{x}$



Metropolis algorithm

- Select initial parameter vector $\mathbf{x}_0$
- Iterate as follows: at iteration number k

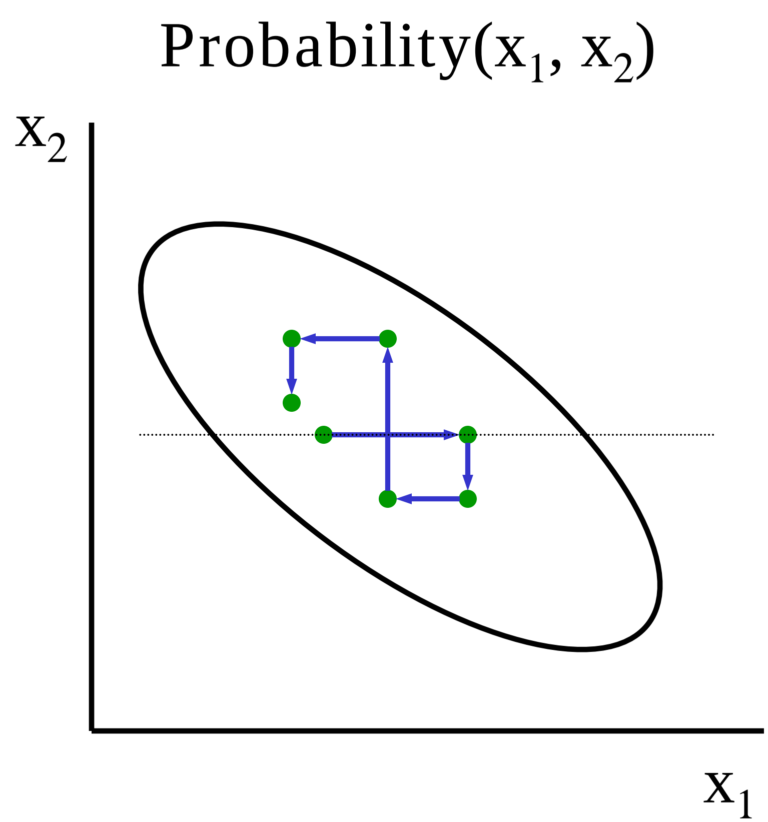
  (1) create new trial position $\mathbf{x}^* = \mathbf{x}_k + \Delta\mathbf{x}$,
  where $\Delta\mathbf{x}$ is randomly chosen from $t(\Delta\mathbf{x})$

  (2) calculate ratio $r = q(\mathbf{x}^*)/q(\mathbf{x}_k)$

  (3) accept trial position, i.e. set $\mathbf{x}_{k+1} = \mathbf{x}^*$
  if $r \geq 1$ or with probability r, if $r < 1$
  otherwise stay put, $\mathbf{x}_{k+1} = \mathbf{x}_k$
- Only requires computation of $q(\mathbf{x})$ (with arbitrary normalization)
- Creates Markov chain since $\mathbf{x}_{k+1}$ depends only on $\mathbf{x}_k$

Gibbs algorithm

- Vary only one component of **x** at a time
- Draw new value of x_j from conditional pdf
  $q(x_j|\, x_1\, x_2 \ldots\, x_{j-1}\, x_{j+1} \ldots\,)$
- Cycle through all components



Hybrid MCMC method

- Called hybrid method because it alternates Gibbs & Metropolis steps (better called "Hamiltonian" method?)
- Associate with each parameter x_i a momentum p_i
- Define a Hamiltonian (sum of potential and kinetic energy):

$$H = \varphi(\mathbf{x}) + \Sigma\, p_i^2/(2\, m_i)\ ,$$

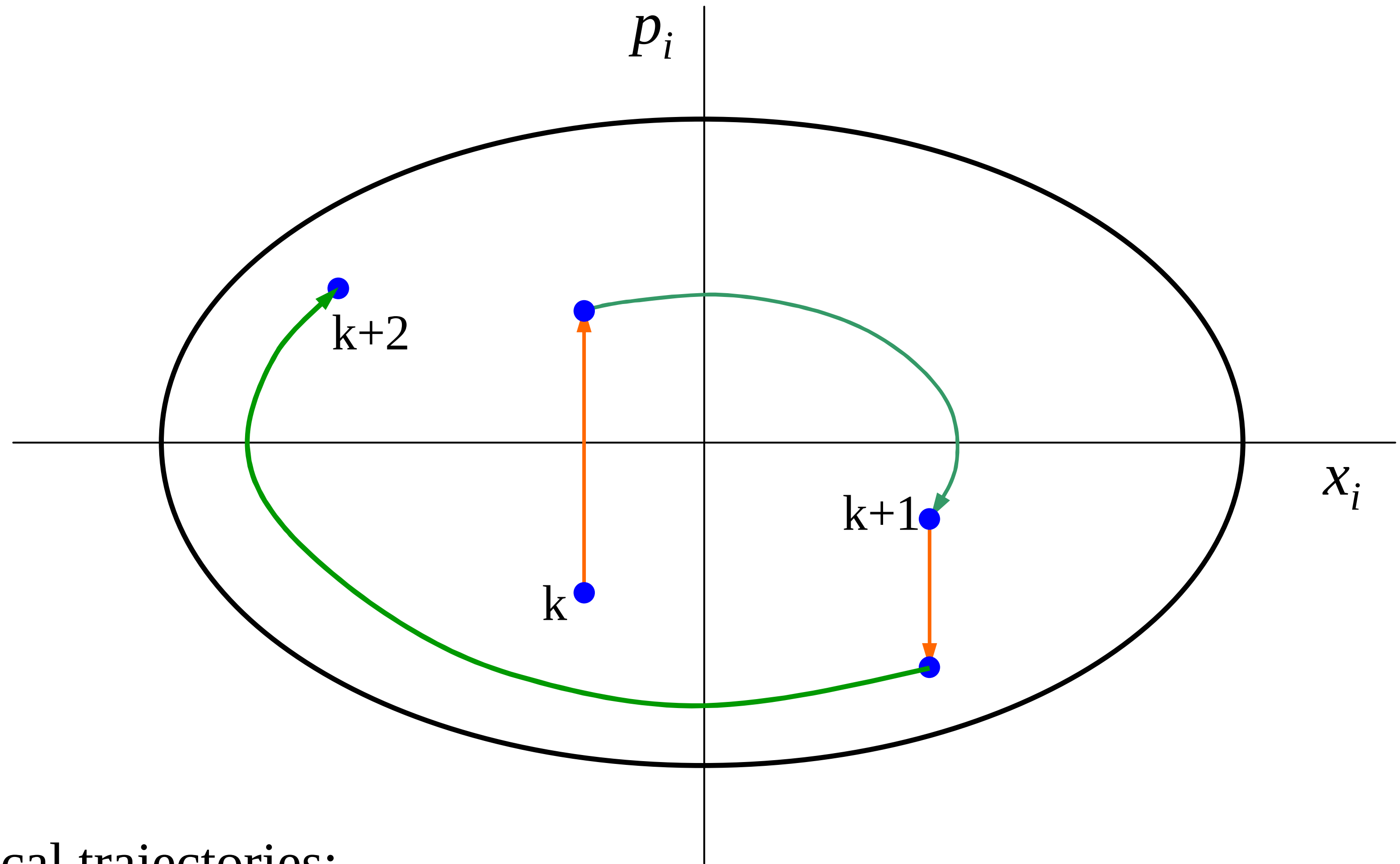
where $\varphi = -\log\,(q(\mathbf{x}))$

- Objective is to draw samples from new pdf:

$$q'(\mathbf{x}, \mathbf{p}) \propto \exp(-\, H(\mathbf{x}, \mathbf{p})) = q(\mathbf{x})\, \exp(-\Sigma\, p_i^2/(2\, m_i))$$

- Samples $\{\mathbf{x}_k\}$ from $q'(\mathbf{x}, \mathbf{p})$ represent draws from $q(\mathbf{x})$ because $\mathbf{p}$ dependence marginalized out

Hybrid algorithm



Typical trajectories:

red path - Gibbs sample from momentum distribution

green path - trajectory with constant *H*, follow by Metropolis

Hamiltonian algorithm

- Gibbs step: randomly sample momentum distribution
- Follow trajectory of constant *H* using leapfrog algorithm:

$$p_i(t+\frac{\tau}{2}) = p_i(t) - \frac{\tau}{2}\frac{\partial\varphi}{\partial x_i}\bigg|_{\mathbf{x}(t)}$$

$$x_i(t+\tau) = x_i(t) + \frac{\tau}{m_i}\,p_i(t+\frac{\tau}{2})$$

$$p_i(t+\tau) = p_i(t+\frac{\tau}{2}) - \frac{\tau}{2}\frac{\partial\varphi}{\partial x_i}\bigg|_{\mathbf{x}(t+\tau)}$$

where τ is leapfrog time step.

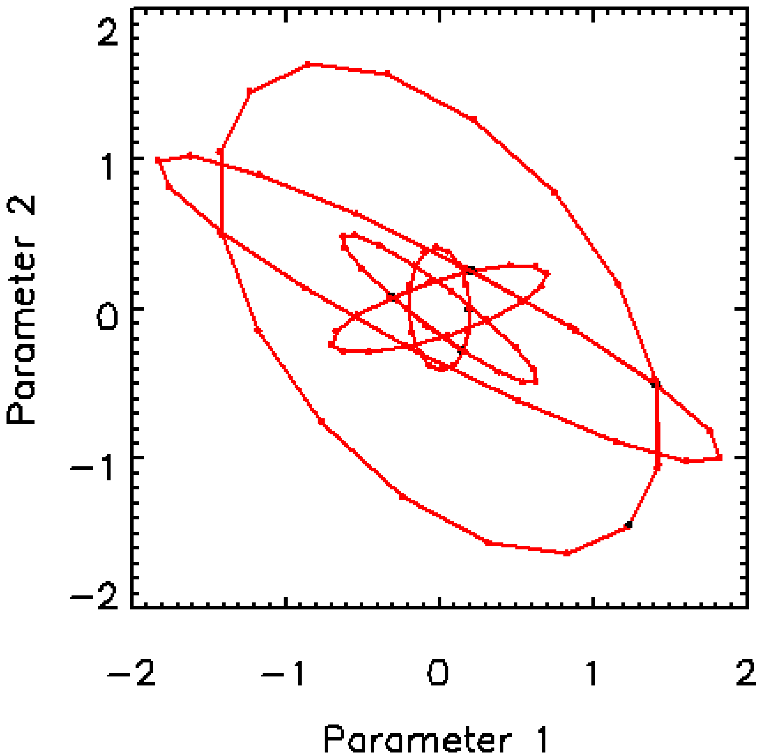
- Repeat leapfrog a predetermined number of times
- Metropolis step: accept or reject on basis of *H* at beginning and end of H trajectory

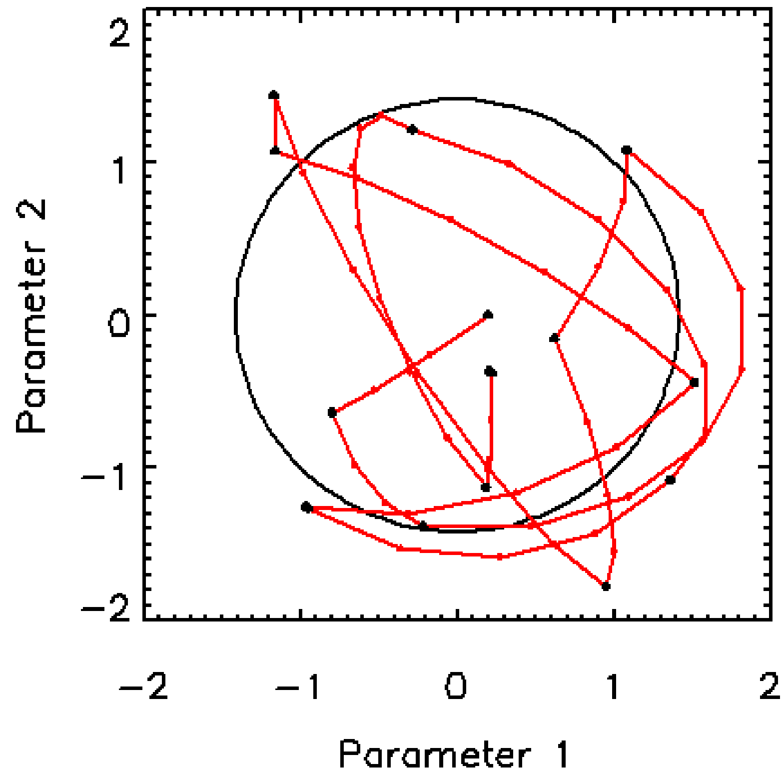
Hybrid algorithm implementation

- Gibbs step - easy because draws are from uncorrelated Gaussian
- H trajectories followed by several leapfrog steps permit long jumps in (**x**, **p**) space, with little change in H
  - specify total time = T ; number of leapfrog steps = T/τ
  - randomize T to avoid coherent oscillations
  - reverse momenta at end of H trajectory to guarantee that it is symmetric process (condition for Metropolis step)
- Metropolis step - no rejections if H is unchanged

- Adjoint differentiation efficiently provides gradient

2D isotropic Gaussian distribution

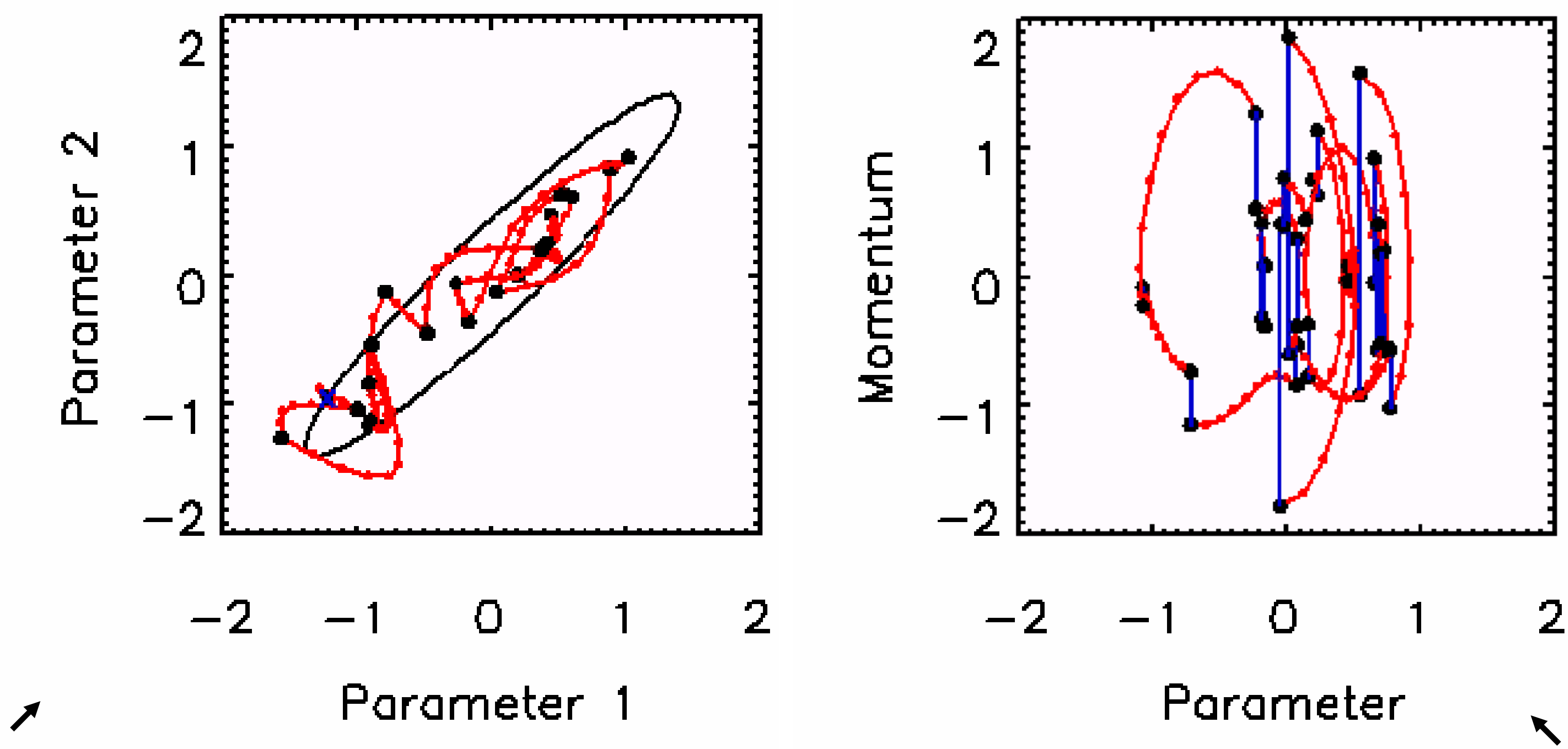




Long H trajectories - shows ellipses when $\sigma_1 = \sigma_2 = 1$, $m_1 = m_2 = 1$

Randomize length of H trajectories to obtain good sampling of pdf

2D correlated Gaussian distribution



- 2D Gaussian pdf with high correlation (r =0.95)
- Length of H trajectories randomized

MCMC Efficiency

– Estimate of a quantity from its samples from a pdf $q(v)$

$$\tilde{v} = \frac{1}{N_k} \Sigma\, v_k$$

– For N independent samples drawn from a pdf, variance in estimate:

$$\mathrm{var}(\tilde{v}) = \frac{\mathrm{var}(v)}{N}$$

– For N samples from an MCMC sequence with target pdf $q(v)$

$$\mathrm{var}(\tilde{v}) = \frac{\mathrm{var}(v)}{\eta\, N}$$

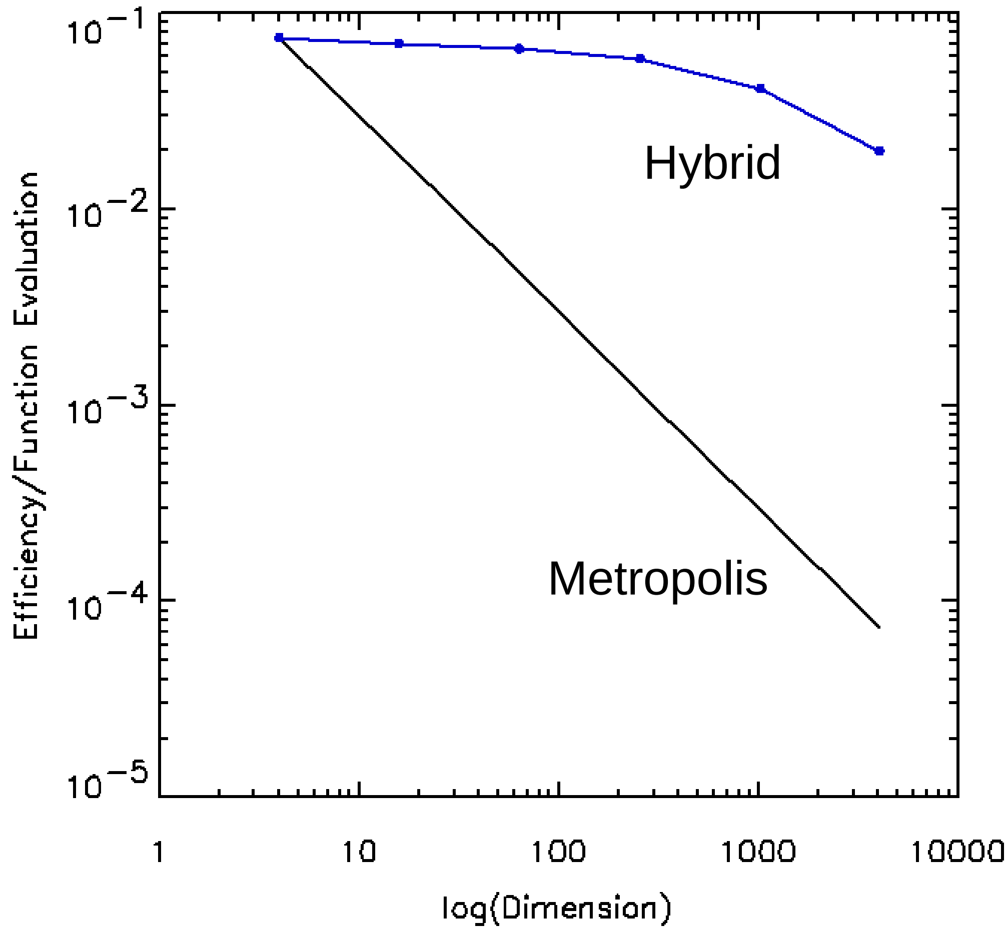
where η is the sampling efficiency

– Thus, η^{-1} iterations needed for one statistically independent sample

– Let v = variance because aim is to estimate variance of target pdf

n-D isotropic Gaussian distributions

- MCMC efficiency versus number dimensions
  - Hamiltonian method: drops little
  - Metropolis method: goes as 0.3/*n*
- Hybrid (Hamiltonian) method much more efficient at high dimensions
- Assumes gradient eval. costs same as function



A new convergence test statistic

- Variance integral

$$\mathrm{var}(x_i) = \int (x_i - \bar{x}_i)^2 \, p(\mathrm{x}) d\mathbf{x}$$

$$= \int \frac{1}{3}(x_i - \bar{x}_i)^3 \nabla \varphi(\mathbf{x}) \, p(\mathbf{x}) \, d\mathbf{x} + \frac{1}{3}(x_i - \bar{x}_i)^3 \, p(\mathbf{x}) \Big|$$

by integration by parts and $\varphi(\mathbf{x}) = -\log(p(\mathbf{x}))$

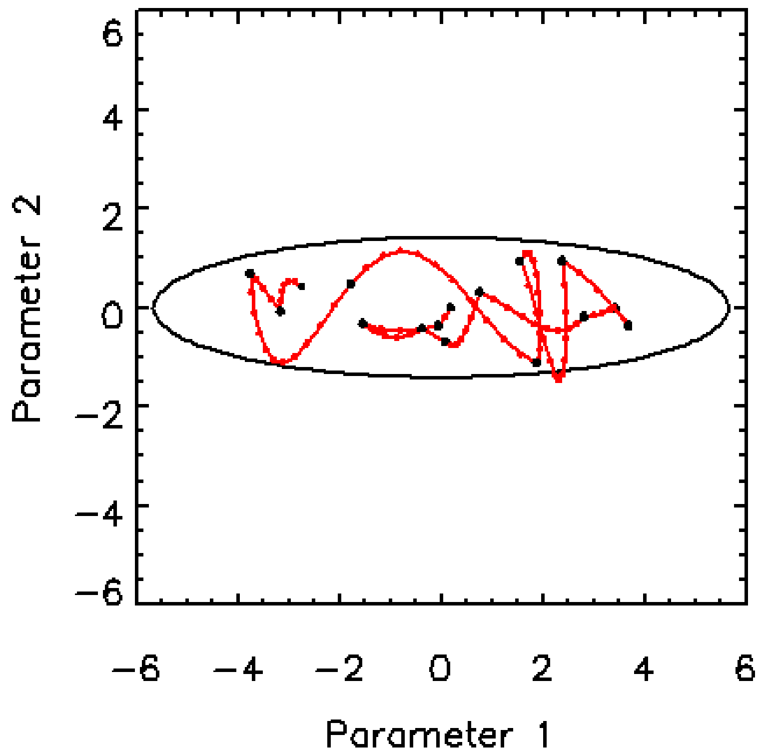
  - limits are typically ±∞ and last term is usually zero
  - thus, integrals are equal

- Form ratio of integrals, computed from samples $\mathbf{x}^k$ from $p(\mathbf{x})$

$$R = \frac{\sum (x_i^k - \bar{x}_i^k)^3 \frac{\partial \varphi}{\partial x_i^k}}{3 \sum (x_i^k - \bar{x}_i^k)^2} \quad , \quad \bar{x}^k = \sum x_i^k$$

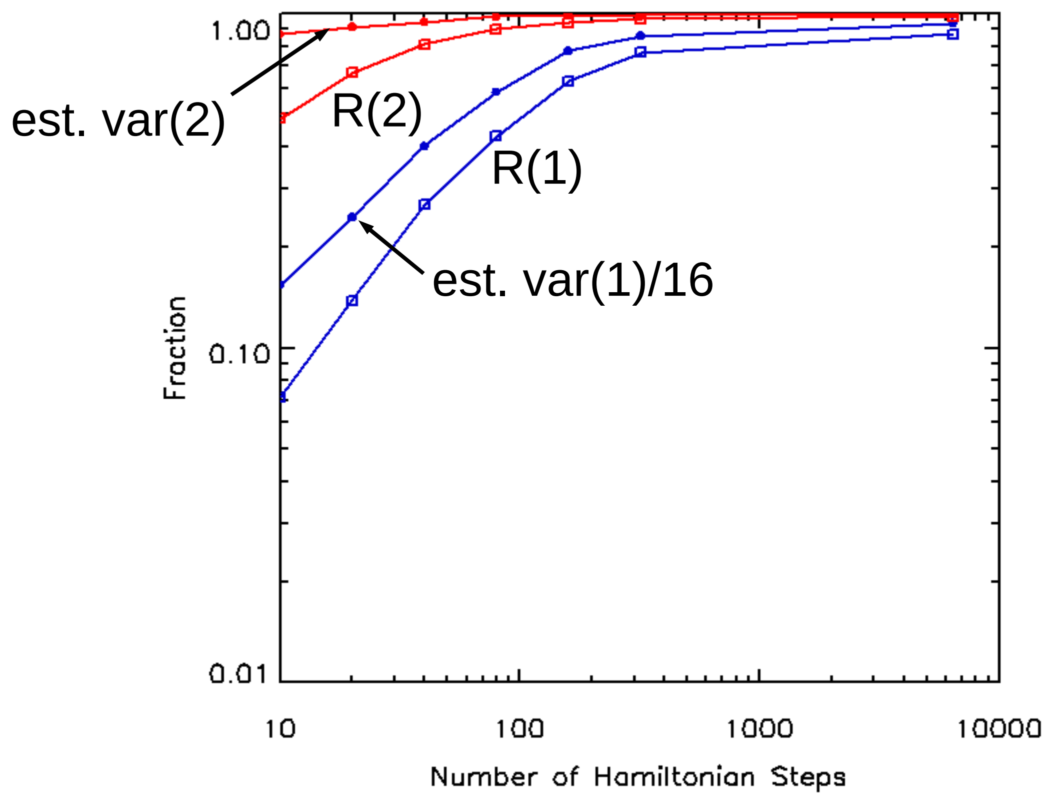
- R tends to be less than 1 when $p(\mathbf{x})$ not adequately sampled

2D non-isotropic Gaussian distribution



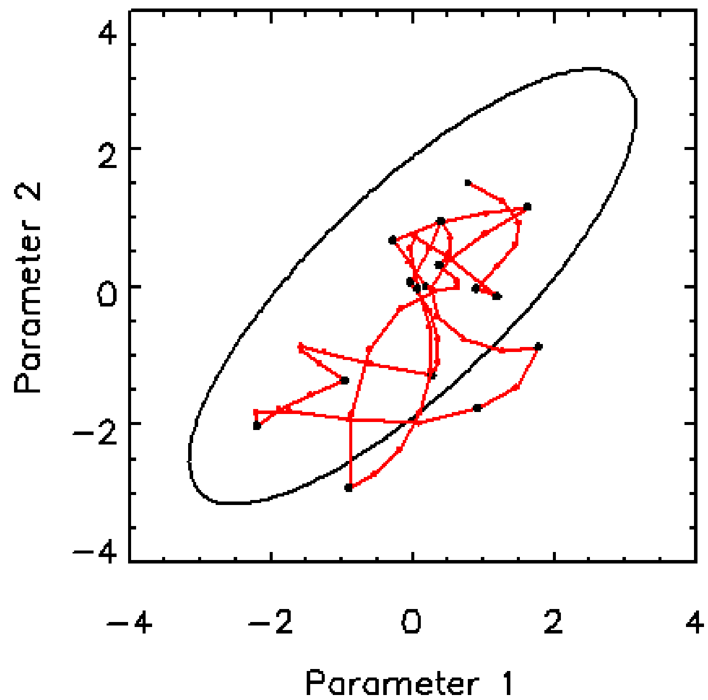
- Nonisotropic Gaussian target pdf: $\sigma_1 = 4$, $\sigma_2 = 1$, $m_1 = m_2 = 1$
- Randomize length of H trajectories to get random sampling
- Convergence; does sequence actually sample target pdf?

Convergence - 2D nonisotropic Gaussians



- Non-isotropic Gaussian target pdf: $\sigma_1 = 4$, $\sigma_2 = 1$, $m_1 = m_2 = 1$
  - control degree of pdf sampling by using short leapfrog steps ($\tau = 0.2$) and $T_{max} = 2$
- Test statistic R < 1 when estimated variance is deficient

16D correlated Gaussian distribution



- 16D Gaussian pdf related to smoothness prior based on integral of L2 norm of second derivative
- Efficiency/(function evaluation) =

  2.2% with Hybrid (Hamiltonian) algorithm

  0.11% or 1.6% with Metropolis; w/o & with covar. adapt.

MCMC - Issues

- Identification of convergence to target pdf
  - is sequence in thermodynamic equilibrium with target pdf?
  - validity of estimated properties of parameters (covariance)
- Burn in
  - at beginning of sequence, may need to run MCMC for awhile to achieve convergence to target pdf
- Use of multiple sequences
  - different starting values can help confirm convergence
  - natural choice when using computers with multiple CPUs
- Accuracy of estimated properties of parameters
  - related to efficiency, described above

Conclusions

- Adjoint differentiation provides efficient calculation of gradient of scalar function of many variables
  - optimization (regression)
  - MCMC, especially hybrid method (other possible uses exist)
- Hybrid method
  - based on Hamiltonian dynamics
  - efficiency for isotropic Gaussians is about 7% per function evaluation, independent of number of dimensions
  - much better efficiency than Metropolis for large dimensions provided gradient can be efficiently calculated
- Convergence test based on gradient of -log(probability)
  - tests how well MCMC sequence samples full width of target pdf

Bibliography

- *Bayesian Learning for Neural Networks*, R. M. Neal, (Springer, 1996); Hamiltonian hybrid MCMC
- "Posterior sampling with improved efficiency," K. M. Hanson and G. S. Cunningham, *Proc. SPIE* **3338**, 371-382 (1998); Metropolis examples; includes introduction to MCMC
- *Markov Chain Monte Carlo in Practice*, W. R. Gilks et al., (Chapman and Hall, 1996); excellent review; applications
- "Bayesian computation and stochastic systems," J. Besag et al., *Stat. Sci.* **10**, 3-66 (1995); MCMC applied to image analysis
- "Inversion based on complex simulations," K. M. Hanson, *Maximum Entropy and Bayesian Methods*, G. J. Erickson et al., eds., (Kluwer Academic, 1998); describes adjoint differentiation and its usefulness
- *Automatic Differentiation of Algorithms*, G. Corliss et al., ed. (Springer, 2002); papers from SIAM workshop on code differentiation
- *Evaluating Derivatives*, A. Griewank (SIAM, 2000); excellent hands-on tutorial on code differentiation

This presentation available under http://www.lanl.gov/home/kmh/